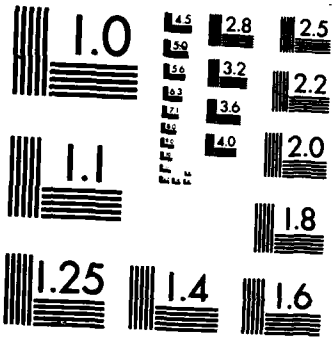
AD-A174 748 A CAD METHOD AND ASSOCIATED ARCHITECTURES FOR LINEAR CONTROLLERS(U) STANFORD UNIV CA INFORMATION SYSTEMS LAB S P BOYD ET AL 1986 N00014-86-K-0112 1/1

UNCLASSIFIED F/G 9/2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD-A174 748

1986 Allerton Proceedings

A CAD Method and Associated Architectures for Linear Controllers

S. P. Boyd [1], V. Balakrishnan, C. H. Barratt,
N. M. Khraishi, X. Li, D. G. Meyer, S. A. Norman
Information Systems Laboratory
Department of Electrical Engineering
Stanford University
Stanford, CA 94305

DTIC
ELECTE
DEC 4 1986
S D

Abstract— New architectures and an associated CAD method are proposed for linear controllers. The architecture is suggested by recent results which parametrize all controllers that stabilize a given plant.

With this architecture, the design of controllers is a convex programming problem which can be solved numerically even for large systems. Constraints on the closed-loop system such as asymptotic tracking, decoupling, limits on peak excursions of variables, step response settling time and overshoot, as well as frequency domain inequalities of any H_∞ variety are readily incorporated in the design. The minimization objective is general and can include LQG, H_∞, new ℓ_1 types, or any combination thereof.

To specify the constraints and objectives mentioned above, we are developing a *control specification language*. This control specification language will be the input to a *compiler* which will translate the specifications into a standard convex program in $\mathbb{R}^L$. A small but powerful subset of the language has been specified and its associated compiler implemented. The results are very encouraging.

The new architecture not only makes *design* of the controller simple but also its *implementation*. These controllers can be built right now from off the shelf components or integrated into silicon using standard VLSI cells.

Details will appear soon in a full length paper.

I. Basic Setup

The basic plant we consider is shown in Figure 1. We decompose its input into two signal vectors, w, the *exogenous inputs*, and u, the *control* or *actuator inputs*. Its outputs we decompose into y, the *measured* or *sensor outputs*, and z, the *regulated variables*. Our job is to design a controller K, with input y *only*, and output u.

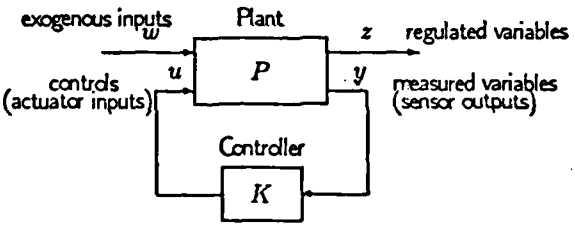


Figure 1: Basic plant and controller.

The signal y represents the signal actually accessible to the controller K, including any command inputs, which may be considered exogenous inputs (i.e. components of w) passed directly to some of components of y.[2] Similarly the signal u represents those inputs to the plant which our controller may vary, that is, those inputs to the plant manipulable by the controller. Thus it is by *definition* that the controller has input y only and output u only.

The exogenous input, w, will include real physical disturbances or noises (torques, forces) acting on the plant, any actuator or sensor noise, and, as mentioned above, any

[1] Research supported in part by ONR under N00014-86-K-0112 and NSF under ECS-85-52465.

[2] We may think of the exogenous inputs which represent commands as the actual physical commands inputs, e.g. angle of a potentiometer, and the corresponding components of y as the 'sensed command', e.g. voltage output from a potentiometer.

DTIC FILE COPY

86 12 02 0

command inputs. w may also contain fictitious inputs injected anywhere in the plant.

The signal z represents any system signal about which we will express a specification, regardless of its accessibility to the controller. Obvious examples are the actual positions, forces, temperatures, etc. we wish to regulate or control. z may include 'internal states' or variables we wish to limit. Some components can be fictitious, e.g. linear combinations of state variables or even filtered versions of signals. z may also include components of u or y.

Thus H_{zw}, the multi-input multi-output closed-loop map from w to z, contains, by definition, every *closed-loop* map of interest.

To illustrate this decomposition of inputs and outputs, we present a simple example. A motor with shaft encoder is used to regulate the angle, θ, of an arm to some (small) commanded angle, θ_{ref}, while a disturbance torque, d, acts on the arm; see Figure 2. Let

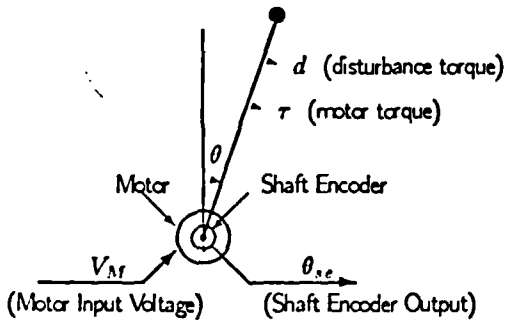


Figure 2: Example system.

θ_{se} denote the shaft encoder output, and let $n_{sens} = \theta - \theta_{se}$, the difference between the actual angle and the sensed angle, that is, sensor quantization 'noise'. Exogenous inputs are

$$w = \begin{bmatrix} \theta_{ref} \\ d \\ n_{sens} \end{bmatrix}.$$

There are two sensed outputs,

$$y = \begin{bmatrix} \theta_{ref} \\ \theta_{se} \end{bmatrix}.$$

Note that the command θ_{ref} is simply passed through P; thus the block diagram of Figure 1 could be drawn in a more conventional way as a two degree of freedom SISO controller.

There is only one actuator input, the motor input voltage, V_m; that is, $u = V_m$. A possible choice of regulated variables is

$$z = \begin{bmatrix} \theta \\ \theta - \theta_{ref} \\ \tau \\ V_m \end{bmatrix}$$

where τ is the motor torque. We will refer to this example several times in the sequel.

We assume only that the I/O map of the plant, P, is linear. For the moment, we do not say whether the system is continuous or discrete time, since much of the ensuing discussion is independent of this. In any case, a real system is almost certainly hybrid with a continuous time physical plant and a discrete, possibly multirate, controller.

We partition the map P as

$$P = \begin{bmatrix} P_{zw} & P_{zu} \\ P_{yw} & P_{yu} \end{bmatrix}$$

so that

$$\begin{bmatrix} z \\ y \end{bmatrix} = P \begin{bmatrix} w \\ u \end{bmatrix}.$$

Of course we have

$$H_{zw} = P_{zw} + P_{zu}K(I - P_{yu}K)^{-1}P_{yw} \quad (1)$$

where H_{zw} is the closed-loop map from the exogenous inputs w to the regulated variables z.

Note that H_{zw} depends on the controller K in a linear fractional fashion. A relatively simple constraint on H_{zw}, e.g. that a certain entry be zero, corresponds via (1) to a much more complicated constraint on the controller K. We remark here that if $P_{yu} = 0$ then (1) simplifies considerably to

$$H_{zw} = P_{zw} + P_{zu}KP_{yw}.$$

Here H_{zw} is *affine* in K. Note that this corresponds to the case where there is essentially *no feedback* in the system.

II. Parameterization of Stabilizing Controllers

In this section we discuss the recent parametrization of all stabilizing controllers. Precise definitions and all details can be found in Vidyasagar's book [1].

A basic requirement is that the controller stabilize the plant. A recent major theme in control theory is that the set of H_{zw}'s achievable with controllers which stabilize the system is a *linear variety*, that is, a translation of a linear subspace.

A linear variety, $\mathcal{L}$, in a vector space, $\mathcal{V}$, is often described as the nullspace or range of an affine map from or to $\mathcal{V}$. Thus we might have

$$\mathcal{L} = \{v \in V \mid Av = b\}$$

where A is a linear map from $\mathcal{V}$ to $\mathcal{W}$ and $b \in \mathcal{W}$. This is a description of $\mathcal{L}$ in terms of a *linear equality constraint*; if $\mathcal{W}$ is $\mathbb{R}^k$, then we can interpret each entry of $Av = b$ as a single linear functional equality constraint in $\mathcal{V}$. Let $\mathcal{H} = \{H_{zw} \mid \text{system is closed-loop stable}\}$, the set of achievable H_{zw}'s. $\mathcal{H}$ may be described via linear equality constraints using the interpolation conditions [6]

Alternatively we may describe a linear variety, $\mathcal{L}$, as the range of an affine map:

$$\mathcal{L} = \{Cu + d \mid u \in \mathcal{U}\}$$

where C is a linear map from $\mathcal{U}$ to $\mathcal{V}$ and $d \in \mathcal{U}$. Such a representation can be regarded as a *free parametric representation* of $\mathcal{L}$; u is a free parameter.

A free parametric representation of $\mathcal{H}$ can be derived using stable coprime fractional theory [6,1] One standard form is the 'Q-parametrization':

$$\mathcal{H} = \{T_1 + T_2 Q T_3 \mid Q \text{ stable}\}.$$

where T_1, T_2, and T_3 are some stable maps.

Note that the Q-parametrization

$$H_{zw} = T_1 + T_2 Q T_3 \tag{2}$$

has exactly the same form as the no-feedback formula mentioned in Section 1; we will soon see a block diagram interpretation of the Q-parametrization in which the parameter Q 'sees' no feedback. In particular, H_{zw} is affine in Q.

The derivation of the Q-parametrization starts with any controller, K_o, which stabilizes the plant and which we will call the *nominal controller*. Let $K_o = Y^{-1}X$ be a left stable coprime factorization [1] of the nominal controller and $P_{yu} = \tilde{D}^{-1}\tilde{N}$ be a left stable coprime factorization of P_{yu}. Then every controller of the form

$$K = (Y - Q\tilde{N})^{-1}(X + Q\tilde{D}),\ Q \text{ stable}$$

stabilizes P and conversely every controller which stabilizes P has this form for some stable Q. There is a similar characterization of the stabilizing controllers in terms of right coprime factorizations.

Since K_0 stabilizes P_{yu}, there is a right coprime factorization of P_{yu}. with $XN + YD = I$ = the identity map. A little calculation yields the Q-parametrization formula, $H_{zw} = T_1 + T_2 Q T_3$, with

$$T_1 = P_{zw} + P_{zu} D X P_{yw},$$

$$T_2 = P_{zu} D,$$

$$T_3 = \tilde{D} P_{yw}.$$

A simple block diagram of the Q-parametrization is shown in Figures 3 and 4.

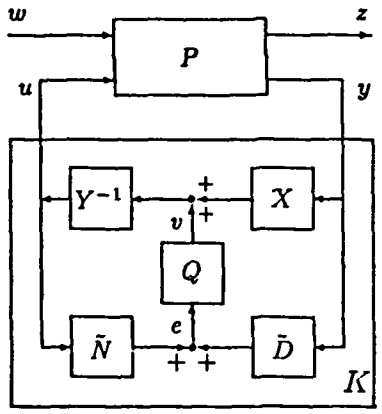


Figure 3: Block diagram of Q-parametrization.

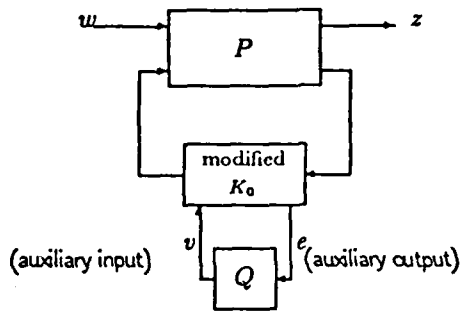


Figure 4: Q-parametrization as modification to nominal controller.

T_1 is simply the H_{zw} achieved with the nominal controller K_o; T_2 is the map from w to e, T_3 is the map from v to z (see Figure 4). The key to the parametrization is that the closed-loop map from v to e is *zero*, so that Q 'sees' no feedback.

Doyle [7] has given a very nice interpretation of the Q-parametrization when the nominal controller is an estimated state feedback. In this case e is simply the output prediction error, $\hat{y} - y$, of the observer, and v is just added before the observer tap to the output of the nominal controller as shown in Figure 5. The controller shown in Figure 5 is sometimes

called an *observer-based controller* (OBC); the point is that *every* controller which stabilizes P can be realized as an observer-based controller.

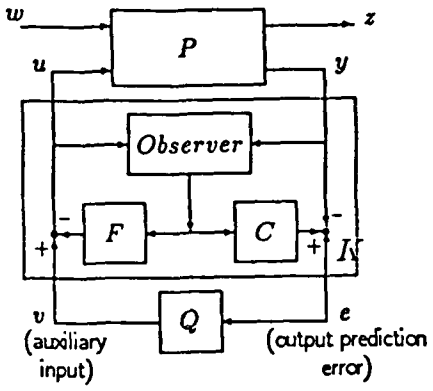


Figure 5: Doyle's interpretation of Q-parametrization for estimated state feddbcak nominal controller. Here F is a stabilizing state feedback gain and $y = Cx$ where x is the plant state.

III. Closed-Loop Constraints and Objectives

In this section and the next, we consider the problem of designing the parameter Q, given the nominal maps T_1, T_2, and T_3. We first observe that many typical requirements on the *closed-loop* performance of our system result in *convex* constraints on H_{zw} and thus on the parameter Q. Before proceeding, we note that requirements on the *open-loop* system generally do *not* result in convex constraints on H_{zw} or Q. One important example is the requirement that the controller, K be diagonal, that is, that K be a decentralized controller. Another important example is the requirement that the controller be stable.

Suppose our design problem is specified as

$$\begin{array}{ll} \text{minimize} & \Phi(H_{zw}) \\ H_{zw} \in \mathcal{K} & \\ \text{cl. loop stable} & \end{array}$$

where Φ is a convex functional and $\mathcal{K}$ a convex (constraint) set. This is equivalent to

$$\begin{array}{ll} \text{minimize} & \tilde{\Phi}(Q) \\ Q \in \tilde{\mathcal{K}} & \end{array} \tag{3}$$

where

$$\tilde{\mathcal{K}} = \{Q \mid T_1 + T_2 Q T_3 \in \mathcal{K}\}$$

and

$$\tilde{\Phi}(Q) = \Phi(T_1 + T_2 Q T_3).$$

Note that $\tilde{\Phi}$ and $\tilde{\mathcal{K}}$ are convex. Of course $\tilde{\mathcal{K}}$ may be empty; this simply means that *no stabilizing controller* can satisfy the constraint $H_{zw} \in \mathcal{K}$.

We now list some typical constraints (and objectives) on H_{zw}, some collection (sum) of which might describe the constraint set $\mathcal{K}$ (objective functional Φ).

For purposes of discussion, we assume the system is discrete-time. H will denote the transfer matrix of H_{zw}, h its impulse response matrix, and $s(t) = \sum_{i=0}^{t} h(i)$ its step response. Of course a convex constraint or functional of H, h, or s is a convex constraint or functional of H_{zw}.

Where possible, we will refer to the example system of §1, which we presume has been sampled at some appropriate rate. When referring to this system, we will use *symbolic* subscripts written in square brackets so the reader need not refer to §1 to find which components of w and z have which interpretations. Thus we will write $H[\theta][\theta_{ref}]$ instead of H_{11}.

- *Asymptotic Tracking, Decoupling, and Regulation*

  The step response from some command input to the regulated variable it is supposed to control must converge to one, e.g.

  $$\lim_{t\to\infty} s[\theta][\theta_{ref}](t) = 1.$$

  Equivalently,

  $$H[\theta][\theta_{ref}](e^{j0}) = 1.$$

  This constraint is a single linear functional equality constraint on H, hence, of course, a convex constraint.

  Asymptotic tracking of ramps or more complicated inputs can be handled as two or more linear functional equality constraints, e.g. $H[\theta][\theta_{ref}](e^{j0}) = 1$, $H'[\theta][\dot{\theta}_{ref}](e^{j0}) = 0$.

  Asymptotic regulation and *asympototic decoupling* are similar constraints. We may require that a regulated variable asymptotically reject constant inputs appearing at certain exogeneous inputs. When the exogenous input is another command input, this constraint is asymptotic decoupling; when the exogenous input is some disturbance, this constraint is asymptotic regulation. For example, to ensure that θ asymptotically

rejects any constant disturbance torque, we might specify

$$H[\theta - \theta_{ref}][d](e^{j0}) = 0.$$

- *Overshoot, Undershoot, and Settling-Time Limits*

  We may require that some step response lie in the unshaded region of Figure 6, e.g.

$$-0.3 \le s[\theta][\theta_{ref}](t) \le 1.3; \qquad 0 \le t \le 10,$$
$$|s[\theta][\theta_{ref}] - 1(t)| \le 1/(t+1); \qquad t > 10.$$

  This constraint can be expressed as a collection of linear functional ineqqualities on s (hence H_{zw}): $L(t) \le s_{32} \le U(t)$ for $t = 0, 1, \ldots$. Thus the set of H_{zw}'s satisfying this constraint is *convex* (it may of course be empty).

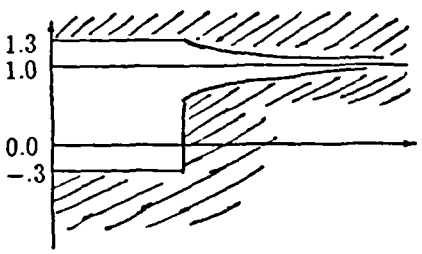


Figure 6: Step response overshoot, undershoot, and settling time constraints.

- *Bounds on Closed-Loop Signal Peaks*

  Given bounds, W_j, on each exogenous input, we may require that each regulated variable be bounded by some given maximum Z_i. This constraint could arise from the requirement not to saturate an actuator or sensor or exceed some internal variable force, torque, or current limit. This constraint is equivalent to

$$\sum_{j=1}^{N_{exog}} W_j \sum_{t=0}^{\infty} |h_{ij}(t)| \le Z_i \text{ for } i = 1, \ldots N_{reg}$$

  where N_{exog} is the number of exogenous inputs and N_{reg} the number of regulated variables.

  Thus the constraint

$$0.1 \sum_{t=0}^{\infty} |h[V_m][\theta_{ref}](t)|$$
$$+ 0.03 \sum_{t=0}^{\infty} |h[V_m][d](t)|$$
$$+ 0.001 \sum_{t=0}^{\infty} |h[V_m][d](t)| \le 50$$

  will guarantee that whenever the command input θ_{ref} is bounded by 0.1, the disturbance d by 0.03, and the sensor noise n_{sens} by 0.001, the motor voltage input V_m will be bounded by 50.

  Again, this is a convex constraint on h, and thus on H_{zw}.

- *Small RMS Disturbance Response*

  If the exogenous input, w, is driven by a wide-sense stationary stochastic process with some specified spectral distribution, then $\mathrm{E}z(t)^T G^T G z(t)$, where G is some weighting matrix, should be as small as possible. Typically, the stochastic process would have zero power in the exogeneous input channels which are command inputs, and G would have only a few nonzero entries.

  We have

$$\mathrm{E}z(t)^T G^T G z(t) = \frac{1}{2\pi} \times \text{the Trace of:}$$

$$G \int_0^{2\pi} H(e^{j\Omega}) S_w(e^{j\Omega}) H(e^{-j\Omega})^T d\Omega \, G^T$$

  where S_w is the power spectral density of the stochastic process driving w. This objective is a *nonnegative quadratic functional* of H and thus of H_{zw}, in particular it is convex.[3]

  For our example, a possible S_w would be

$$S_w(e^{j\Omega}) = \begin{array}{c} \begin{bmatrix} 0 & 0 & 0 \\ 0 & S_d(e^{j\Omega}) & 0 \\ 0 & 0 & \Delta^2/12 \end{bmatrix} \end{array} \begin{array}{l} (\theta_{ref}) \\ (d) \\ (n_{sens}) \end{array}$$

  where $S_d(e^{j\Omega})$ is the spectral density of the disturbance torque and Δ is the step size of the shaft encoder. Note that we put no noise power in the command input θ_{ref}. If we are interested in the noise power in the pointer angle, θ, then we let $G = [\,1\ 0\ 0\ 0\,]$ and the functional above is simply

$$\mathrm{E}\theta^2 = \frac{1}{2\pi} \times$$
$$\left[\int_0^{2\pi} |H[\theta][d](e^{j\Omega})|^2 S_d(e^{j\Omega})\, d\Omega \right.$$
$$\left. + \int_0^{2\pi} |H[\theta][n_{sens}](e^{j\Omega})|^2 \Delta^2/12\, d\Omega \right]$$

[3] Here *quadratic functional* includes a *linear functional* and a *constant (functional)* term.

which can be interpreted as the total noise power in θ due to the disturbance and sensor noise.

- *Bounds on Transfer Function Peak Magnitudes*

We may specify an upper bound for some entry (or block of entries) of H, e.g. $|H_{65}(e^{j\Omega})| \leq U(\Omega)$ for all Ω, where $U(\Omega)$ is some bound function. This constraint may arise in several ways.

First, many classical specifications of bandwidth or peaking are expressed this way. Referring to our example, we might specify

$$|H[\theta-\theta_{ref}][\theta_{ref}](e^{j\Omega})| \leq 0.03 \text{ for } |\Omega| \leq \Omega_B$$

so that for frequencies below Ω_B, we get -30db of rejection.

$$|H[\theta-\theta_{ref}][\theta_{ref}](e^{j\Omega})| \leq 3 \text{ for all } \Omega$$

to keep gain in the command to tracking error map under about 10db for all frequencies.

The second way a transfer function peak magnitude constraint can arise is as follows. If we require a small RMS response of some regulated variable, as in the previous constraint, but know only a bound on the total power in the process driving the exogenous input w, and not its spectral distribution, then we are led to a transfer function peak magnitude constraint; this is the basis of H_∞ control theory.

Third, a *sufficient* condition to stabilize not just the plant, but all plants 'near' our given plant (in a peak frequency response deviation sense), may be formulated as a bound on certain closed-loop frequency response magnitudes.

These H_∞ type constraints are convex constraints on H and hence H_{zw}.

- *Miscellaneous Bounds and Objectives*

We mention here some less common constraints. A *slew-rate* constraint on the step response, say s_{11}, may be enforced as:

$$|h_{11}(t)| \leq R \text{ for all } t.$$

A *passivity* or *minimum dissipation* constraint, say on the 1, 1 entry of H_{zw}, can be formulated as

$$\Re H_{11}(e^{j\Omega}) \geq -D \text{ for all } \Omega.$$

Such a constraint can be used, for example, to guarantee stability of the system with a saturating actuator. All constraints involving step responses can be generalized to other, arbitrary, input signals, or even sets of signals. Thus we may specify a maximum peak tracking error for command inputs bounded by B_{cmd} and slew rate under R_{cmd}.

Finally we mention that the boundary between constraint and objective is not sharp; we could, for example, try to minimize an overshoot in a certain step response, or put hard constraints on the RMS response.

At this point it should be clear that a large number of practical constraints on, and objectives for, closed-loop system performance can be formulated as a standard convex program for the parameter Q.

This program for Q is infinite dimensional and generally cannot be solved analytically except in special cases. The small RMS disturbance response objective alone can be solved using Wiener-Hopf or LQG methods; certain forms of the small peak transfer function objective problem are solved by H_∞-optimal control theory; certain forms of the small peak disturbance objective problem are solved by the new l_1-optimal control theory introduced by Vidyasagar [8] and developed by Dahleh and Pearson [2].

IV. Specifications to Convex Program Compiler

The translation from closed-loop constraints or objective on H_{zw} to convex program for the parameter Q is mathematically straightforward. As mentioned above, the resulting program is infinite dimensional. An approximate solution can be found numerically when the parameter Q is restricted to a large, but finite, dimensional space. Let

$$Q = \sum_{i=1}^{L} x_i \Gamma_i$$

where $x_i \in \mathbf{R}$ and Γ_i are fixed stable maps. We will call $x = [x_1, ..., x_L]^T$ the *decision variables*. With this additional restriction, we have a standard finite dimensional convex program in $\mathbf{R}^L$

$$\begin{array}{ll} \text{minimize} & f(x) \\ x \in \hat{\mathcal{K}} & \end{array} . \qquad (4)$$

While the convex constraint set $\hat{\mathcal{K}}$ cannot be explicitly found, it is easily approximated, and then this explicit finite dimensional convex program can be solved numerically by any of a number of methods.

Let us first briefly discuss the choice of parametrization and the Γ_i's, since this affects how well the finite dimensional program (4) approximates the program (3).

In theory, it makes no difference which nominal controller nor which coprime factorization is used to form the T_i's in the Q-parametrization; all parametrizations produce precisely the same set, $\mathcal{H}$, of H_{zw}'s. The *particular* Q which yields a given H_{zw}, however, will depend on the choice of T_i's. As mentioned above, the Q's will be restricted to some finite dimensional space, so the set of H_{zw}'s achievable *in practice* is a finite dimensional linear variety containing T_1, which is the H_{zw} achieved with the nominal controller. This means that the nominal controller used should yield at least reasonable performance, so that we have more of a chance of finding a 'moderate' Q which achieves good performance. By 'moderate' we mean a Q which is neither too 'big' nor too 'stiff', that is, has too broad a spread of dynamics. An inappropriate nominal stabilizing controller will place severe demands on the filter Q. For example, a state feedback which places the eigenvalues far to the left of the bandwidth achievable when other factors such as control authority are taken into account, will require a 'large' Q filter, essentially to cancel the unreasonably large plant input signal generated by the excessive state feedback. So in practice, the choice of nominal controller is not arbitrary.

A similar comment applies to the choice of coprime factorizations, which will affect the maps T_2 and T_3. Roughly speaking, they should be of reasonable size and have most of their dynamics not too far from the final closed-loop bandwidth.

We turn now to the practical problem of translating a large number of constraints/objective on step responses, transfer functions, RMS responses, and so on, into the corresponding convex program for the decision variables. We propose that this job be mechanized by the use of a *compiler* which accepts as input a list of closed-loop constraints and a description of the objective and as output produces the convex program for the decision variables. The constraints and objective would be specified in a *control specification language*, which would be natural for the control engineer, refering directly to step responses, noise powers, transfer functions, etc. The output of the compiler would be a convex program which could be solved by a convex program solver, as shown in Figure 7.

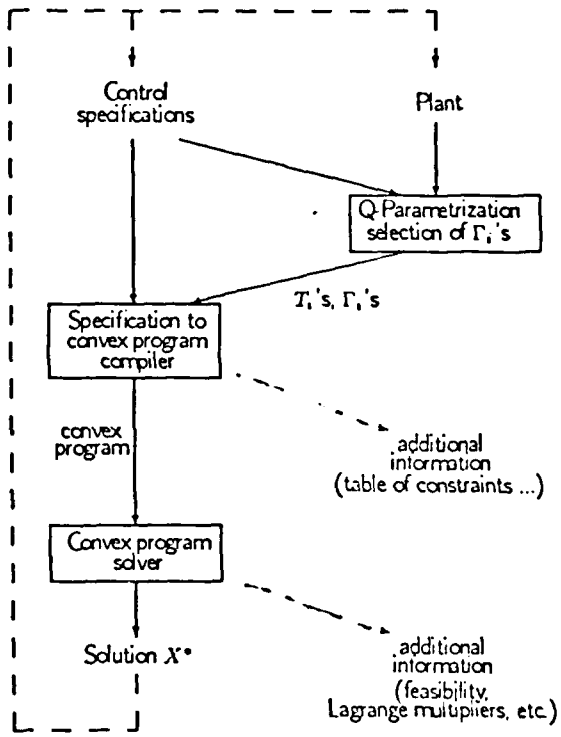


Figure 7: Design of approximately optimal controller using control specification to convex program compiler.

IV.A Control Specification Language

Here we discuss the *control specification*, the input to the compiler. Many of the features we discuss have actually been implemented in our compiler; §VI describes what we have implemented so far.

An overall general structure for the control specification is

```
constraints {
   <constraint>;
   ...
   <constraint>;
```

}

```
objective {
   description of objective
}
```

For example, the constraints might have the forms encountered in §III, such as

constraints {
$H[0][0_{ref}](e^{j0}) = 1;$
$-0.3 \leq s[0][0_{ref}](t) \leq 1.3 \quad \text{for } 0 \leq t \leq 10;$
$0.1 \sum_{t=0}^{\infty} |h[V_m][0_{ref}](t)|$
$+0.03 \sum_{t=0}^{\infty} |h[V_m][d](t)|$
$+0.001 \sum_{t=0}^{\infty} |h[V_m][d](t)| \leq 50;$
$\Re H_{11}(e^{j\Omega}) \geq -D$ for all Ω;
}

A description of the objective could be

objective {
$\frac{1}{2\pi} \int_0^{2\pi} |H[0][d](e^{j\Omega})|^2 S_d(e^{j\Omega})\, d\Omega$
$+ \frac{1}{2\pi} \int_0^{2\pi} |H[0][n_{sens}](e^{j\Omega})|^2 \frac{\Delta^2}{12}\, d\Omega$
}

For a detailed description of the syntax, see §VI.

V. Two Proposed Controller Architectures

Let $Q^* = \sum_{i=1}^{L} x_i^* Q_i$ be a satisfactory Q for the control problem. The resulting controller

$$K^* = (Y - Q^*\tilde{N})^{-1}(X + Q^*\tilde{D})$$

will generally be *full*, (each input will affect every output) and have a large number of states, perhaps hundreds. The standard procedure at this point would be to apply some sort of model reduction to arrive at a low order controller, K_{red}, with only tens of states. We propose instead two methods for implementing the full K^*, both based on the Q-parametrization. Both methods rely on the availability and ease of VLSI implementation of many-tap FIR filters, used extensively in signal processing, for example in equalizers for modems. Thus, our controllers will tend to have far greater orders than is currently common. While it is certain that there is a much lower order K_{red} which provides satisfactory performance, we question the need to find it.

V.A Modified Nominal Controller Architecture

Consider the block diagram of the Q-parametrization given in Figure 4. We propose that controllers *actually be built with the architecture shown in Figure 4*, with Q realized (possibly) separately in hardware as an FIR filter. This architecture can be thought of as a simple modification of the nominal controller; the nominal controller must be modified to yield an auxiliary output signal e and accept an auxiliary input signal v. In some controllers, the signal e is easily available, even though Figure 3 suggests that we need an additional $\tilde{N}$ and $\tilde{D}$. Similarly, injecting the signal v as shown in Figure 4 is also straightforward.

For example, if the nominal controller is an estimated state feedback, then as mentioned earlier, e is the output prediction error of the observer, and v is just added to the output of the nominal controller, before the observer tap, as shown in Figure 5. Thus, in many cases the hardware and/or computational costs of modifying the nominal controller are slight, since the Q filter does all of the additional processing.

We note that this method can be used to improve *existing* controllers.

V.B FIR Coprime Architecture

Generally, the nominal controllers which provide easy access to auxiliary output e are those with complexity on the order of the plant, the estimated state feedback nominal controller a good example. The reason is, roughly speaking, that e is the pseudo-state observer error, which means that the modified nominal controller really contains a pseudo-state observer within it, which is likely to have the same complexity as the plant. This is admittedly vague, and only offered as our interpretation.

In nominal controllers of much lower complexity than the plant, e.g., a diagonal PI controller, e is usually not easily accessible, and must be reconstructed. In this case we propose that the entire controller be implemented using FIR filters.

Consider the coprime factorization of the controller K^* given above. Suppose Y, Q^*, $\tilde{N}$, X, and $\tilde{D}$ are all FIR, or very nearly approximated by FIR filters. Recall that unlike the plant or controller, these operators are *stable* and hence their impulse matrices decay to zero. With proper choice of sample rate and parametrization, 100 tap FIR filters should be more than adequate. In this case, K^* can be

realized as a feedback connection around one FIR filter, as we now show. K^* is governed by the equations

$$Yu - Xy = v = Q\tilde{N}u + Q\tilde{D}$$

(see Figure 3). We rewrite these equations as

$$u = (I - V)u + Xy + v$$

$$v = Q\tilde{N}u + Q\tilde{D}y$$

which we interpret as an FIR filter, F, with input u, y and v, and output u and v, and

$$F = \begin{bmatrix} I - V & X & I \\ Q\tilde{N} & Q\tilde{D} & 0 \end{bmatrix}.$$

This is shown in Figure 8. Of course there are

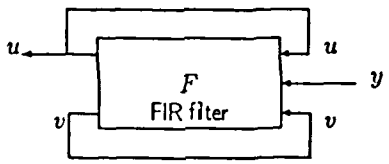


Figure 8: FIR coprime architecture for controllers.

many other F's which realize K^*, and we do not yet know a good procedure for picking one.

Finally, we mention that for both architectures, it is possible to vary the effect of the parameter Q from 'off' to 'on' by inserting a gain, α, which varies between zero and one, in the right place (e.g. in the bottom loop of Figure 8). Thus we have a *homotopy* (continuous deformation) of our nominal controller K_0 into our designed controller K^*, with all intermediate controllers stabilizing the plant, indeed with H_{zw} affine in the parameter α. Such a parameter might be quite useful as a *control authority parameter.* If our nominal controller has somewhat lower authority/bandwidth than our designed controller, but is more robust to plant variations or failures, then α can vary the control authority and robustness of our controller from nominal (resp., low and high) to designed (resp., high and low). In an emergency, we can smoothly drop back to the nominal controller. Of course, this will only work with moderate plant variations, and cannot take the place of a real fault tolerant control system (with fault detection and reconfiguration).

VI. Implementation of a Subset

We have implemented a compiler which recognizes a small but powerful subset of *control specification language.* Time and frequency domain linear and magnitude frequency inequality constraints, and quadratic objective are recognized. The system must have only one actuator, and the filter Q is simply a FIR filter. We are currently implementing a much more general compiler.

We assume Q has n_{iq} inputs and *a single output*; so far we have implemeted a compiler for control problems that have only one actuator or control input. Q is assumed to be an n_{coeff}-tap FIR filter in every channel. Thus, there are a total of $n_{iq} \times n_{coeff}$ parameters in the filter Q and these are the *decision variables.* In this case

$$H_{zw} = T_1 + \sum_{i=1}^{n_{iq}} Q_i G_i$$

where Q_i is the map from the ith input of Q to its output and

$$G_i = T_2 e_i^T T_3$$

(e_i is the *ith* unit vector). Since Q is FIR with n_{coeff} taps in each channel, the system impulse response at a given discrete time k is

$$h_{zw}(k) = t_1(k) + \sum_{i=1}^{n_{iq}} \sum_{j=0}^{n_{conv}} q_i(j) g_i(k-j),$$

where

$$n_{conv} \triangleq \min(n_{coeff} - 1, k).$$

Clearly the system impulse response at any discrete instant, k, is affine in the decision variables, $q_i(j)$.

Our compiler recognizes three time domain functionals:

$$ustep[i][j](k),$$
$$h[i][j](k),$$
$$q[i](k).$$

These are, respectively, the step and impulse responses from input j to output i at time k, and the step response of Q from it's input i at time k (recall that Q has a single output).

Three frequency domain functionals are recognized:

$$Re_H[i][j](r,\theta),$$
$$Im_H[i][j](r,\theta),$$
$$Mag_H[i][j](r,\theta).$$

These are, respectively, the real part, imaginary part, and magnitude of the frequency response from input j to output i at $re^{j\theta}$, r and

θ are real. Since our current convex programming package allows only linear constraints, constraints involving $MagH$ are parsed into two contraints on the real and imaginary parts of H— at the expense of at most a 1.5dB error. It could be modified to parse into more constraints, yielding less error.

Real scalar variables and scalar expressions involving standard mathematical functions are allowed.

Functional inequalities are allowed to take the form

$$f \leq e;$$

$$f \geq e;$$

$$f = e;$$

$$e \leq f \leq e;$$

$$|f| \leq e;$$

where f represents one of the above functionals and e represents any scalar expression.

A quadratic objective can be specified as the sum of any number of the following terms:

$$h_sqr[i][j](k),$$
$$Mag_H_sqr[i][j](r,\theta),$$
$$q_sqr[i](k).$$

These are, respectively, the square of the impulse response from input j to output i at time k, the square of the frequency response magnitude from input j to output i at $re^{j\theta}$, and the square of the impulse response of Q from its input i to it's output at time k.

Nested looping is allowed. This makes repetitive constraints and objective terms easy to specify, for example

```
constraints {
for t = 0 to 10
   - 0.3 ≤ ustep[0][θref](t) ≤ 1.3;
}
objective {
for Ω = 0 to π step 0.03
   0.03 * Mag_H_sqr[0][d](e^{jΩ});
}
```

The standard C language preprocessor cpp is used, which allows the use of macros and file include facilities.

VII. Acknowledgements

Various preliminary forms of this material were presented at seminars at Berkeley, USC, Cal Tech, and Honeywell SRC; several people made substantial suggestions (which we took): C. A. Desoer, J. Doyle, M. Safonov, P. Kokotovic, and R. Kosut.

References

[1] M. Vidyasagar, *Control System Synthesis: A Factorization Approach*, MIT Press, 1985

[2] M.A. Dahleh and J.B. Pearson, "ℓ_1 Optimal feedback controllers for discrete-time systems." Technical report. TR 8513. Rice University, Houston, TX 77251 Sept. 1985

[3] G. Zames, "Feedback and optimal sensitivity: Model reference transformations, multiplicative seminorms, and approximate inverses." *IEEE Trans. Automat. Contr.*, vol AC-26, pp. 301-320, 1981

[4] J.W. Helton, "An H_∞ approach to control." in *Proc. CDC* pp. 607-611, Dec. 1983

[5] B.C. Chang and J.B. Pearson, "Optimal disturbance rejection in linear multivariable systems." *IEEE Trans. Automat. Contr.*, vol AC-29, pp. 880-887, Oct. 1984

[6] C.A. Desoer, R.W. Liu, J. Murray, and R. Sacks, "Feedback system design: The fractional representation approach to analysis and synthesis." *IEEE Trans. Automat. Contr.*, vol AC-25, pp. 399-412, June. 1980

[7] J.C. Doyle, "Matrix interpolation theory and optimal control." Ph.D Thesis, University of California, Berkeley 1984

[8] M. Vidyasagar, "Optimal rejection of persistent bounded disturbances." *IEEE Trans. Automat. Contr.*, vol AC-31, pp. 527-535, June 1986

A1

END

1-87

DTIC